J Sched (2011) 14: 141–156
DOI 10.1007/s10951-009-0115-y

On robust online scheduling algorithms

Michael Gatto · Peter Widmayer

Published online: 21 July 2009

Abstract While standard parallel machine scheduling is concerned with good assignments of jobs to machines, we aim to understand how the quality of an assignment is affected if the jobs' processing times are perturbed and therefore turn out to be longer (or shorter) than declared. We focus on online scheduling with perturbations occurring at any time, such as in railway systems when trains are late. For a variety of conditions on the severity of perturbations, we present bounds on the worst case ratio of two makespans. For the first makespan, we let the online algorithm assign jobs to machines, based on the non-perturbed processing times. We compute the makespan by replacing each job's processing time with its perturbed version while still sticking to the computed assignment. The second is an optimal offline solution for the perturbed processing times. The deviation of this ratio from the competitive ratio of the online algorithm tells us about the "price of perturbations". We analyze this setting for Graham's algorithm, and among other bounds show a competitive ratio of 2 for perturbations decreasing the processing time of a job arbitrarily, and a competitive ratio of less than 2.5 for perturbations doubling the processing time of a job. We complement these results by providing lower bounds for any online algorithm in this setting. Finally, we propose a risk-aware online algorithm tailored for the possible bounded increase of the processing time of one job, and we show that this algorithm can be worse than Graham's algorithm in some cases.

Keywords Robustness · Online · Scheduling · Graham's Algorithm · Perturbation

1 Introduction

Optimization problems are often solved for input data which only estimate reality. Practical examples include shortest path and optimum tour computations in graphs when edge costs represent experienced travel times: The optimum solution defines a travel plan, but executing the plan in reality will most likely lead to a real travel time that differs from the planned time, due to unforeseeable (smaller or larger) disturbances, caused by the real traffic situation at the moment of travel. In general, the quality of the computed plan can suffer from these perturbations when the plan is executed, and in extreme cases, the computed plan can even become infeasible for the perturbed real data.

In some settings, a computed plan can be adjusted in realtime, when a perturbation is observed. A car driver may adjust his trip according to the travel congestion news in the radio, and a railway operator may reassign a train coming into a station with a delay to a proper, free track. In this paper, we limit ourselves to situations in which such an adjustment is not possible. In the railway example, for instance, there may not be a free track at the desired point in time, or there may not be a chance to let waiting passengers go to a different platform, so that the delayed train needs to wait for its assigned track to become free.

Generally speaking, the solution for the original instance of the problem can, alas, not be changed after the perturbations are revealed. Our interest is in the effect of these

This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority—6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

M. Gatto · P. Widmayer (✉)
ETH Zurich, Zurich, Switzerland
e-mail: widmayer@inf.ethz.ch

M. Gatto
e-mail: gatto@inf.ethz.ch

perturbations on the quality of the precomputed solution, in the worst case. More specifically, we want to compare the quality of the plan, evaluated on the perturbed data, with the quality of the plan based on the original data. In this paper, we take a step in this direction by studying the parallel machine scheduling problem. As a first step, we aim to study how Graham's algorithm (Graham 1966) performs with respect to perturbations. Our motivation for choosing this algorithm is twofold: On the one hand, the algorithm is popular and very well understood, so it can nicely serve to illustrate the principles at work when perturbations occur; and on the other hand, it works online, emphasizing that planning decisions may be irrevocable. Technically, the input to the parallel machine scheduling problem is a sequence of processing times for n jobs, each of which needs to be processed later on one of m identical machines. A feasible solution assigns each job to exactly one of the machines. The objective is to minimize the completion time of the job that terminates latest (the *makespan*) (Pinedo 2002). In *online* parallel machine scheduling, jobs are presented in consecution, and a job must be assigned to a machine at the time of its presentation (Sgall 1998). Just like in the other examples above, there is a clear distinction between the presentation of the processing times of the jobs in the planning phase, and processing the jobs on the planned machines, that is, executing the plan.

Consider the following example shown in Fig. 1: We need to schedule online on two machines the sequence of jobs with processing times $\frac{1}{2}$, 1 and $\frac{1}{2}$ using Graham's algorithm. The algorithm assigns the two jobs of processing time $\frac{1}{2}$ on one machine, and the job of processing time 1 alone on one machine. This is the plan obtained by Graham's algorithm on the unperturbed instance, and has makespan 1, which is optimal. Now, assume that the execution time of one of the jobs of processing time $\frac{1}{2}$ turns out to be $\frac{3}{2}$. By substituting the original processing time with this perturbed processing time in the plan previously obtained by Graham's algorithm, we obtain a makespan of 2. The optimum offline solution with the perturbed processing times assigns the job of processing time $\frac{3}{2}$ on one machine, and the remaining two jobs the other machine, and achieves the optimal makespan of $\frac{3}{2}$. For this example, the ratio between the latter two makespans is $\frac{4}{3}$, and we are interested in a worst-case analysis of this ratio across all inputs.

Obviously, if the job processing times of the plan have nothing at all to do with the times during execution, nothing of interest can be said about the quality of the plan. We therefore focus on situations that bound perturbations in some way. More specifically, we aim at understanding to what extent a *limited number of perturbations of the processing times* affects the performance of online parallel machine scheduling. We assume that a probability distribution of perturbations is either not available, or does not tell a lot (locomotive engine breakdowns are an example, since they can occur anywhere with a very small probability), and focus on a worst-case analysis.

We study the behavior under perturbations of the well-known algorithm by Graham (1966) for online scheduling, the *List scheduling algorithm*. Graham's algorithm assigns the next presented job to the machine that will terminate earliest for the job sequence seen so far. Although online algorithms with better competitive ratio exist, Graham's algorithm remains, in its simplicity and with its competitive ratio of $2 - \frac{1}{m}$, a prime example of an interesting online algorithm. All the perturbations of the processing times are disclosed after the entire solution has been determined. This models the situation in which the entire plan is produced before its execution starts (note that if each job's perturbed processing time would be revealed immediately after it has been assigned to a machine, Graham's algorithm would not be misled at all). We measure the impact of perturbations as the worst-case ratio of two makespans. The first makespan is the makespan that we get by taking Graham's assignment of jobs to machines for the original instance, and by replacing, within this assignment, all original processing times with the perturbed ones. The second is the makespan of an optimal offline solution of the perturbed instance. This ratio naturally tends to be larger than Graham's competitive ratio; the larger it is, the more Graham's algorithm suffers from perturbations. Our performance measure is not applicable to all online problems, since a solution computed for the unperturbed instance might not be feasible for the perturbed instance (this happens if, for example, the problem has some resource constraint which can be violated by the perturbation, such as in bin packing). For parallel machine scheduling, however, any assignment of jobs to machines is feasible.

This setting can also be interpreted as introducing a more powerful adversary with respect to the well-known online setting. Not only must each decision be taken immediately each time the adversary presents a new slice of the problem instance, but the adversary may deliberately distort the information he gives. Naturally, the extent of the distortion greatly affects the performance of online algorithms which

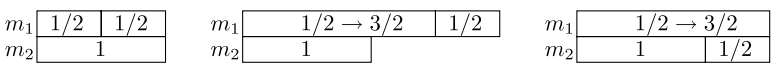

Fig. 1 The simple example with three jobs and two machines: *Left*, the plan obtained online with Graham's algorithm on the unperturbed instance. *Center*, the same plan applied to the perturbed processing times. *Right*, an optimal offline solution for the perturbed processing times

are not tuned for distortions. It is therefore no surprise that Graham's scheduling algorithm performs better for bounded perturbations than for unbounded ones.

1.1 Related work

Robustness has been defined in a variety of ways in optimization theory. Exact input values have been replaced by probability distributions (Scharbrodt et al. 2006) in a probabilistic approach, or by intervals of possible values in a worst case scenario (Montemanni and Gambardella 2004; Kasperski and Zieliński 2006), or by uncertainty sets (Ben-Tal and Nemirovski 2002). A related but different question to our view of robustness has been addressed in *sensitivity analysis*. Sensitivity analysis asks for the amount of perturbation that can be tolerated before the structure of an optimal solution changes. Sensitivity analysis has been studied predominantly for linear programming (Chvátal 1983), but also for many combinatorial problems such as network flows (Ahuja et al. 1993) and scheduling (Hall and Posner 2004). We point out that an instance which is unstable with respect to sensitivity analysis does not necessarily need to be a bad instance in our worst-case setting. Indeed, although the structure of the optimal solution might change with a very small perturbation, the quality of the solution of the unperturbed optimum can remain very near to the one of the optimum of the perturbed instance for this perturbation. Throughout this paper, we allow a small number of processing times to be perturbed a lot, but we will not change the structure of a solution and merely observe its change in quality because the perturbation is supposed to happen after the irrevocable assignment decision.

Online scheduling on m identical parallel machines has been studied extensively (Sgall 1998). The competitive ratio of $2 - \frac{1}{m}$ of the first deterministic online algorithm by Graham (1966), the *List* scheduling algorithm, was improved several times over the years, with the best deterministic online algorithm today achieving a competitive ratio of 1.9201 (Fleischer and Wahl 2000). Randomized algorithms have also been widely addressed as, for instance, in Albers (2002), with the currently best known (randomized) competitive ratio of 1.916.

The effect of perturbations in offline scheduling algorithms has been studied under many different points of view (Hall and Posner 2004; Kouvelis and Gang 1997). A sensitivity analysis for different parameters (Hall and Posner 2004) also addresses how perturbations affect the objective, and how to reconstruct an optimal solution, given some perturbations.

A study of single machine scheduling for jobs with priorities and release times allows perturbations of release times and pursues the goal to efficiently reconstruct a feasible schedule (Mauroy et al. 1997).

The scheduling variant where binary precedence relations exist between jobs has been addressed as follows in the context of robustness. Consider two jobs scheduled on two different machines that have a direct precedence relation. Then, there is a communication delay before the second job can start being processed, since the result of the first job must be transferred between the two machines. The uncertainty lies in the actual size of the communication delay, which may be given as an interval of possible values. For this setting, Sanlaville (2005) gives an analysis on which estimates an algorithm should use in order to produce a stable solution. For two machines and restricted precedences, Moukrim et al. (2003) provides an algorithm with an absolute performance guarantee with respect to the optimum.

The effect of perturbed processing times has also been addressed in different ways. Parallel machine scheduling where all jobs' processing times are accurate up to a factor $(1 \pm \varepsilon)$ of a declared value was analyzed in Penz et al. (2001). The quality of any algorithm deteriorates by a factor $\frac{1+\varepsilon}{1-\varepsilon}$ for the makespan, and by $\sqrt{1+\varepsilon}$ for the sum of completion times, for some small ε. We provide a better bound for the makespan, since our analysis exploits the structure of Graham's schedule. The performance of scheduling has also been addressed with respect to processing times drawn from a distribution. With this view, the scheduling algorithm needs to schedule the jobs offline with the knowledge of the distributions only, and the actual realization of the processing times are disclosed in an online fashion after the schedule has been computed. This setting has been addressed in terms of average case analysis for the completion times (Scharbrodt et al. 2006) and of minimization of an objective in expectation, as in Möhring et al. (1999). The stochastic online setting, where jobs are revealed in an online fashion with stochastic processing times, has been thoroughly analyzed for the goal of minimizing the completion times of the jobs (Megow et al. 2006).

From the online perspective and with a probabilistic viewpoint, *smoothed competitive analysis* (Becchetti et al. 2006) provides a performance measure of online algorithms with respect to an optimum offline solution if the input values are perturbed with some specific type of random noise. In the context of preemptive scheduling on a single machine with release times, smoothed competitive analysis has been applied to the multilevel feedback algorithm for minimizing the total flow time of the jobs (Becchetti et al. 2006). For this setting, the authors show that, by smoothing the processing times according to the partial bit randomization model (see, e.g., Schäfer 2004) using a distribution function satisfying certain constraints, they can explain the good practical behavior of the multilevel feedback algorithm.

The tolerance to perturbations of Graham's offline List scheduling algorithm has been addressed both in terms of decrease in quality of the objective (Graham 1966, 1969)

and in the number of different (offline) schedules that arise from perturbing the processing time of one job and for different input sequences (Kolen et al. 1994). The List scheduling algorithm was presented for a more complex setting than parallel machine scheduling, which also included precedence constraints. Graham (1966, 1969) analyzed the effects of relaxing a number of instance-defining parameters as, for instance, the number of machines or the precedence constraints. For each parameter, he analyzed the worst-case ratio of the makespans of the relaxed instance with respect to the non-relaxed instance, for both instances scheduled with List. He showed a worst-case ratio of $2-\frac{1}{n}$ for decreasing the processing times, and similar results were derived for the relaxation of other parameters.

1.2 Summary of results

We derive lower bounds on the competitive ratio of Graham's algorithm for the perturbed schedule for the following scenarios: For integer r, and increasing the processing times of $r \leq n$ jobs arbitrarily, we show a ratio of $(2+r-\frac{r+1}{m})$; for arbitrarily decreasing the processing times of any number of jobs scheduled on $r \leq m$ machines in an optimal offline schedule we show a ratio of $(2+\frac{r-1}{m-r})$; for $x>1, x \in \mathbb{Q}_0^+$, and dividing the processing times of r jobs by a factor x we show a ratio of $(2+\frac{r\cdot(x-1)-1}{m})$; for either dividing or multiplying (but not both) the processing times of an arbitrary number of jobs by a factor $x>1$ we show a ratio of $(1+x-\frac{x}{m})$ and $(1+x-\frac{1}{m})$, respectively. We also give infinite families of examples where the bounds are tight or come close. Our results imply specific bounds for specific cases: If the processing times of three jobs increase arbitrarily, Graham's algorithm is $(5-\frac{6}{m})$-competitive; if the processing time of one job decreases arbitrarily it is 2-competitive; and if jobs may triplicate their processing times, it is $(4-\frac{1}{m})$-competitive.

We also provide simple lower bounds on the competitive ratio for several settings of perturbations. We show that no online algorithm can have a competitive ratio smaller than 2 if the perturbation may decrease the processing time of one job arbitrarily. For the setting where the perturbation may decrease the processing time of one job to a factor at least $\frac{1}{x}, x>1, x \in \mathbb{Q}_0^+$, of its original processing time, we show a lower bound of $2-\frac{2}{x+1}$; for the case of perturbations decreasing the processing time of at most two jobs to a factor at least $\frac{1}{x}$ of their original processing time, we show a lower bound of $\min\{x,2\}$. For the case of the perturbations increasing the processing time of one job to a factor at most 2 of its original processing time, we show a lower bound of $\frac{3}{2}$, and for the case of perturbing two jobs to a factor at most 2 of their original processing time we show a lower bound of 2.

Finally, we propose a Graham-like algorithm designed to be robust with respect to the possibility of an x-fold increase of the processing time of one job, for $x>2, x \in \mathbb{Q}_0^+$. We show that the algorithm is x-competitive if no perturbation arises, and $1+x-\frac{1}{x}$ if a perturbation arises, thus performing much worse than Graham's algorithm. We also give examples which come close to these bounds.

2 Problem setting and notation

The problem is specified as a 3-tuple $(J, P, \tilde{P})$, where J is the online sequence of n jobs to be scheduled on m machines, P and $\tilde{P}$ specify the original processing time $p_j \in \mathbb{Q}_0^+$ and the perturbed processing time $\tilde{p}_j \in \mathbb{Q}_0^+$ of each job $j \in J$, respectively. A *schedule* for the sequence of jobs is an assignment of the jobs to the machines. Graham's algorithm (which we briefly recall at the end of this section) schedules the *original instance* of the online parallel machine scheduling problem, that is, the job sequence J with processing times P, and produces a schedule $\mathrm{List}(J, P)$.

Each problem is characterized by the perturbation against which we analyze robustness. The effect of the perturbation is reflected in the processing times $\tilde{P}$, which may *either* increase *or* decrease; the perturbation may be of arbitrary size, or bounded, for each job, by a factor x of the job's original processing time. The latter setting is motivated by project scheduling, where the extent of the misjudgment of a task's processing time is often linked to the task's difficulty. We refer to the jobs J with processing times $\tilde{P}$ as the *perturbed instance* of the online parallel machine scheduling problem.

We denote by $\mathrm{OPT}(J, \tilde{P})$ the optimal offline schedule of the sequence of jobs J with processing times $\tilde{P}$. In the offline setting, the order in J is irrelevant. The *makespan* of a schedule is the time when the last terminating job finishes being processed. In the literature, the makespan is normally denoted by $C_{\max}$. In our setting, the actual processing times of jobs may differ from the originally given ones. Since this concept is different from the usual one, we introduce a different notation. We denote by $\mathcal{L}(S, P)$ the makespan of the schedule S with processing times P. In this setting, we measure the robustness of Graham's online algorithm by considering the makespan of the schedule $\mathrm{List}(J, P)$ obtained with the original instance, but evaluated on the perturbed processing times $\tilde{P}$. Hence, we evaluate $\mathcal{L}(\mathrm{List}(J, P), \tilde{P})$. We compare this makespan with the makespan $\mathcal{L}(\mathrm{OPT}(J, \tilde{P}), \tilde{P})$ of an optimal offline schedule $\mathrm{OPT}(J, \tilde{P})$. Note that comparing $\mathcal{L}(\mathrm{List}(J, P), \tilde{P})$ with $\mathcal{L}(\mathrm{List}(J, P), P)$ does not provide any useful information, since the total processing time is different for P and $\tilde{P}$: If the processing time of a job increases arbitrarily, *any* algorithm needs to process this job. Moreover, we do not compare $\mathcal{L}(\mathrm{List}(J, P), \tilde{P})$ to $\mathcal{L}(\mathrm{List}(J, \tilde{P}), \tilde{P})$, as such a comparison would reveal little about the absolute quality of the algorithm: Even an algorithm that schedules all jobs on a

single machine (out of the $m \gg 1$ available) would have the excellent ratio of 1 with this comparison, although in absolute terms it is clearly extremely poor.

In this model, each machine processes its assigned jobs without pausing in between. The sum of the processing times of a machine is called *load*. When machine $i \in M$ is finished, it remains idle up to the makespan. We refer to the idle time as $s_i, i \in M$. We call the set of machines which process some perturbed jobs the *affected* machines, and denote them by $M^{\neq}$. Similarly, we call the set of machines which do not process any perturbed jobs *unaffected* machines, and denote them by $M^{=}$.

For perturbations increasing the jobs' processing times, we denote the perturbed processing times as $p_j^{\uparrow}$, $j \in J$, and the set of all increased processing times as $P^{\uparrow}$. Similarly, for decreases we use $p_j^{\downarrow}$, $j \in J$ and $P^{\downarrow}$.

Changing the processing times of the jobs in a schedule also influences the idle times. Therefore, we refer to the idle time resulting after a perturbation as $s_i^{\uparrow}$ for increased processing times and as $s_i^{\downarrow}$ for decreases. In the analysis, we look at the perturbations of the jobs sequentially, in any order. In this way, we can specify the impact of the perturbation of each job on the idle time of each machine. When a job changes its processing time, the subsequent jobs shift accordingly in the schedule. This shift may shorten or lengthen the idle time of various machines. We denote the increase or decrease in idle time on machine $i \in M$ caused by perturbing job j by $\delta_j^i \in \mathbb{Q}$, which may be positive or negative.

We recall Graham's *List* scheduling algorithm for the online parallel machine scheduling problem (i.e., without perturbations). When considering the next job of the online sequence, Graham's algorithm schedules it on a machine having least load given the previous assignments of jobs to machines. As a warm-up for our proofs in later sections, let us recall the classical competitive analysis of Graham's algorithm. For a makespan $\mathcal{L}_{\text{List}}$ obtained with Graham's algorithm, the processing times of the jobs and the idle times of the m machines satisfy $m\mathcal{L}_{\text{List}} = \sum_{j\in J} p_j + \sum_{i=1}^{m} s_i$. Let $\mathcal{L}_{\text{OPT}}$ be the optimal makespan (for unperturbed processing times). As each job must be scheduled non-preemptively by any algorithm, $p_j \leq \mathcal{L}_{\text{OPT}}, \forall j \in J$. Furthermore, $\mathcal{L}_{\text{OPT}} \geq \frac{\sum_{j\in J} p_j}{m}$, since no schedule can do better than distribute the total processing time evenly across all machines. Finally, consider an arbitrary job $\bar{j}$ finishing at the makespan of Graham's schedule. Then, $s_i \leq p_{\bar{j}}, \forall i \in M$, since otherwise Graham's algorithm would have scheduled job $\bar{j}$ on the machine not satisfying the inequality. Furthermore, the machine attaining the makespan has zero idle time. Thus, $m\mathcal{L}_{\text{List}} = \sum_{j\in J} p_j + \sum_{i=1}^{m} s_i \leq m\mathcal{L}_{\text{OPT}} + (m-1)p_{\bar{j}} \leq (2m-1)\mathcal{L}_{\text{OPT}}$. This proof shows a competitive ratio of $2 - \frac{1}{m}$ for Graham's algorithm.

Finally, consider the (perturbed) instances where the perturbation increases the processing times. For these cases, $\mathcal{L}(\text{OPT}(J, P), P) \leq \mathcal{L}(\text{OPT}(J, P^{\uparrow}), P^{\uparrow})$ holds, since the total processing time increases and the maximum processing time of the jobs can also only become larger.

To improve the readability of our proofs, we use the following compact notation for the makespans. Instead of $\mathcal{L}(\text{OPT}(J, P), P)$ we write $\mathcal{L}_{\text{OPT}}$, and instead of $\mathcal{L}(\text{OPT}(J, P^{\downarrow}), P^{\downarrow})$ we write $\mathcal{L}_{\text{OPT}}^{\downarrow}$. Similarly, we write $\mathcal{L}_{\text{List}}^{\downarrow}$ instead of $\mathcal{L}(\text{List}(J, P), P^{\downarrow})$ for Graham's algorithm. The notation for increases in processing times is obtained accordingly.

3 Arbitrary perturbations

In the following, we analyze robustness for arbitrary size perturbations of the processing times of jobs. First, we analyze the case of arbitrary decreases in processing times and then of arbitrary increases.

3.1 Arbitrary decreases in processing time

As a first step, we bound the quality of the solution of any best-possible online algorithm if the processing times may decrease arbitrarily.

Theorem 1 *No algorithm for online scheduling on $m \geq 2$ identical parallel machines on instances where the processing time of one job may decrease arbitrarily can have a competitive ratio smaller than* 2.

Proof Assume such an algorithm exists, and consider the following sequence of jobs, all with processing time 1. First, the adversary sequentially presents m jobs. To be strictly better than 2-competitive, any algorithm must schedule each job on a different machine. Then, the adversary presents a final job, which may be scheduled on any machine. Now, the perturbation affects one job which is scheduled alone on a machine, and decreases its processing time to 0. Thus, the computed schedule has an empty machine, and a makespan of 2. The optimum offline perturbed schedule assigns each of the now m jobs on a different machine, and has a makespan of 1. □

Note that the construction can be extended to any number of jobs by introducing an appropriate number of jobs of processing time $\varepsilon \to 0$ which do not influence the construction and allow for a bound arbitrarily close to 2.

In the following, we derive a bound on the competitive ratio and a worst-case instance which matches the bound for the case where the arbitrary perturbations affect r machines, that is, where r machines of Graham's schedule process jobs whose processing time is perturbed. We have the following theorem:

Theorem 2 *Consider the instances of online scheduling on $m \geq 2$ identical parallel machines where perturbations may decrease the processing times of some jobs arbitrarily. Restricted to these instances, if Graham's algorithm schedules the perturbed jobs on $r < m$ machines, Graham's algorithm is $(2 + \frac{r-1}{m-r})$-competitive, and this bound is best possible. For $r = 1$, Graham's algorithm is optimal.*

Proof Consider Graham's schedule $\mathrm{List}(J, P)$. We refer to the jobs scheduled on the unaffected machines $M^{=}$ as $J^{=}$. Hence, by definition, these jobs do not change their processing time. To estimate the makespan after the perturbation we analyze the schedule of the $m - r$ unaffected machines $M^{=}$. We distinguish two cases: in the first, the makespan $\mathcal{L}^{\downarrow}_{\mathrm{List}}$ is attained by at least one machine in $M^{=}$, while in the second no machine in $M^{=}$ attains it.

For the first case, we let $\bar{j}_\mu$ be a job attaining the makespan after the perturbation on an unaffected machine $\mu \in M^{=}$. In this case, the time spent by the machines $M^{=}$ up to the makespan is given by:

$$\begin{aligned}(m-r)\cdot\mathcal{L}^{\downarrow}_{\mathrm{List}} &= \sum_{j\in J^{=}} p^{\downarrow}_j + \sum_{i\in M^{=}} s^{\downarrow}_i \\ &\leq \sum_{j\in J} p^{\downarrow}_j + (m-r-1)\,p^{\downarrow}_{\bar{j}_\mu} \\ &\leq m\mathcal{L}^{\downarrow}_{\mathrm{OPT}} + (m-r-1)\mathcal{L}^{\downarrow}_{\mathrm{OPT}},\end{aligned}$$

since $J^{=} \subseteq J$ and at most $(m - r - 1)$ machines have some idle time, which due to the workings of Graham's algorithm is smaller than $p^{\downarrow}_{\bar{j}_\mu}$. Thus, the bound follows:

$$\mathcal{L}\big(\mathrm{List}(J,P),P^{\downarrow}\big) \leq \left(2+\frac{r-1}{m-r}\right)\cdot\mathcal{L}\big(\mathrm{OPT}(J,P^{\downarrow}),P^{\downarrow}\big).$$

For the second case, let $\bar{j}_\mu$ be a job attaining the makespan after the perturbation on an affected machine $\mu \in M^{\neq}$. Because of Graham's algorithm, when $\bar{j}_\mu$ was originally scheduled, μ was a machine with the least load, say with load ℓ. Since the processing times on unaffected machines $M^{=}$ remain unchanged, the latter machines have load at least ℓ after the decreases. Thus, their idle time is $s^{\downarrow}_i \leq p^{\downarrow}_{\bar{j}_\mu}, i \in M^{=}$, since $\bar{j}_\mu$ attains the makespan. Note that this bound holds both if $\bar{j}_\mu$ is perturbed or it remains unchanged. Thus, the time spent by the machines in $M^{=}$ up to $\mathcal{L}^{\downarrow}_{\mathrm{List}}$ can be represented as follows:

$$\begin{aligned}(m-r)\cdot\mathcal{L}^{\downarrow}_{\mathrm{List}} &= \sum_{j\in J^{=}} p^{\downarrow}_j + \sum_{i\in M^{=}} s^{\downarrow}_i \\ &\leq \sum_{j\in J} p^{\downarrow}_j - p^{\downarrow}_{\bar{j}_\mu} + (m-r)\,p^{\downarrow}_{\bar{j}_\mu} \\ &\leq m\cdot\mathcal{L}^{\downarrow}_{\mathrm{OPT}} + (m-r-1)\cdot\mathcal{L}^{\downarrow}_{\mathrm{OPT}},\end{aligned}$$

since by the case analysis $\bar{j}_\mu$ is not scheduled on a machine in $M^{=}$. The bounds lead to the same expression as in the previous case, thus concluding the first part of the proof.

A worst-case instance for $r \leq m - 2$ machines has the following structure, illustrated in Fig. 2. First, the adversary presents r huge jobs with processing time $C > 1 + \frac{r-1}{m-r}$ each. Graham's algorithm schedules each on a different machine. Next, the adversary presents $m - r$ big jobs with processing time one. Again, Graham's algorithm schedules one of these jobs on each of the $m - r$ idle machines. Then, the adversary presents $(r - 1) \cdot (m - r)$ small jobs with processing time $\frac{1}{m-r}$. Graham's algorithm schedules these jobs evenly on the machines with load smaller than C, such that each of these machines has load $1 + \frac{r-1}{m-r}$. Finally, a big job with processing time one is presented, and is scheduled on one of the machines with load smaller than C. Now, the perturbation decreases the processing time of all huge jobs from C to zero. In this way, the online algorithm was forced to work with only $m - r$ out of the m available machines, and the achieved makespan is $2 + \frac{r-1}{m-r}$. The optimal offline algorithm, on the other hand, schedules each big job on a different machine, thus using $m - r + 1$ machines, and schedules $m - r$ small jobs on each of the remaining $r - 1$ machines. In this way, it achieves a makespan of 1. For $r = m - 1$, the bound evaluates to $\mathcal{L}^{\downarrow}_{\mathrm{List}}/\mathcal{L}^{\downarrow}_{\mathrm{OPT}} \leq m$. A simple worst-case example matching this bound first presents $m - 1$ big jobs of processing time m, followed by m small jobs of processing time 1. The perturbations shrinks the processing time of the $m - 1$ big jobs to zero. The stated bound easily follows. The analysis above is not suited for $r = m$. Nevertheless, the bound $\mathcal{L}^{\downarrow}_{\mathrm{List}}/\mathcal{L}^{\downarrow}_{\mathrm{OPT}} \leq m$ can be achieved with the example for $r = m - 1$, by additionally scheduling a job of processing time $\frac{1}{m}$ between the big jobs and the small jobs, and by perturbing its processing time to zero as well.

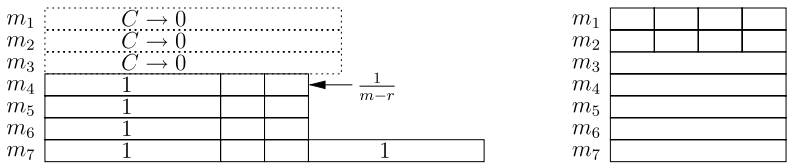

Fig. 2 A worst-case example (for $r = 3$) matching the bound of Theorem 2. *Left*, Graham's schedule on the original instance; the perturbed jobs have a dotted outline. *Right*, the optimal schedule of the perturbed instance

The optimality of Graham's algorithm for $r = 1$ follows from Theorem 1, since the arbitrary decrease in processing time of one job affects one machine. □

Intuitively, this proof shows that the worst-case scenario happens if the affected machines are blocked with jobs whose processing time decreases to zero. Hence, the adversary and the perturbation force the online algorithm to work with r machines less than initially stated. Surprisingly, this affects the competitive ratio only with an additive term of $\frac{r}{m-r}$ with respect to the usual performance of Graham's algorithm. Intuitively, this increase means that the jobs which can be processed on the affected machines in the optimum offline solution are evenly spread on the unaffected machines in Graham's algorithm. Finally, for $r = 0$ we match the bound for Graham's algorithm.

3.2 Arbitrary increases in processing time

We now consider arbitrary increases in the job's processing times. The results show that the worst-case perturbation happens if arbitrarily small jobs are scheduled on the machine attaining the makespan before the perturbation, and the perturbation lets these small jobs increase a lot.

We first consider the case where r jobs increase their processing time arbitrarily. Then, we give a similar analysis for the case where any number of jobs may be perturbed, and the jobs result to be scheduled on r machines of an optimal offline schedule of the perturbed instance. Also in this case, our analysis matches Graham's bound for $r = 0$.

Theorem 3 *Consider the instances of online scheduling on m identical parallel machines where perturbations may increase the processing times of r jobs arbitrarily. Restricted to these instances, Graham's algorithm has a competitive ratio of $2 + r - \frac{r+1}{m}$, and this bound is best possible for $r \leq m - 2$.*

Proof Let $J^{\uparrow} \subset J, |J^{\uparrow}| = r$ be the set of jobs whose processing times increase. Let $\psi_j = p_j^{\uparrow} - p_j$ be the increase of job $j \in J^{\uparrow}$, which can be bounded by $\psi_j \leq p_j^{\uparrow} \leq \mathcal{L}_{\mathrm{OPT}}^{\uparrow}, j \in J^{\uparrow}$. Recall that, for the sake of the analysis, we assume the perturbations to occur sequentially. Thus, each perturbed job has an associated difference in idle time for each machine; hence, $-\psi_j \leq \delta_j^k \leq \psi_j, k \in M, j \in J^{\uparrow}$. Furthermore, at least one machine $\mu \in M$ attains the makespan before the perturbation, and has idle time $s_\mu = 0$. The idle times of the remaining machines satisfy $s_i \leq \mathcal{L}_{\mathrm{OPT}} \leq \mathcal{L}_{\mathrm{OPT}}^{\uparrow}, i \in M$. Furthermore, the increase of job $j \in J^{\uparrow}$ does not increase the idle time of the machine the job is scheduled on (although it may decrease it). The new makespan of the machines is thus given by:

$$
\begin{aligned}
m \cdot \mathcal{L}_{\mathrm{List}}^{\uparrow} &= \sum_{j \in J} p_j + \sum_{i \in M} s_i + \sum_{j \in J^{\uparrow}} \psi_j + \sum_{j \in J^{\uparrow}, k \in M} \delta_j^k \\
&= \sum_{j \in J} p_j^{\uparrow} + \sum_{i \in M} s_i + \sum_{j \in J^{\uparrow}, k \in M} \delta_j^k \\
&\leq m \mathcal{L}_{\mathrm{OPT}}^{\uparrow} + (m-1)\mathcal{L}_{\mathrm{OPT}}^{\uparrow} + r(m-1)\mathcal{L}_{\mathrm{OPT}}^{\uparrow}, \\
\mathcal{L}\big(\mathrm{List}(J, P), P^{\uparrow}\big) &\leq \left(2 + r - \frac{r+1}{m}\right) \\
&\quad \times \mathcal{L}\big(\mathrm{OPT}\big(J, P^{\uparrow}\big), P^{\uparrow}\big).
\end{aligned}
$$

To see that the analysis is best possible for $r \leq m - 2$, we give an example, illustrated in Fig. 3, where this bound is achieved up to an arbitrarily small $\varepsilon \in \mathbb{Q}_0^+, 0 < \varepsilon < \frac{1}{m}$. The adversary uses three types of jobs, namely big, small and tiny jobs. The processing times of the jobs within each class are similar and specified in the following. First, the adversary presents $m - r - 1$ big jobs with processing time $1 - \frac{(r+1)}{m}$. Graham's algorithm schedules each on a different machine. The adversary presents $m(r+1)(1 - \frac{r+1}{m}) - 1$

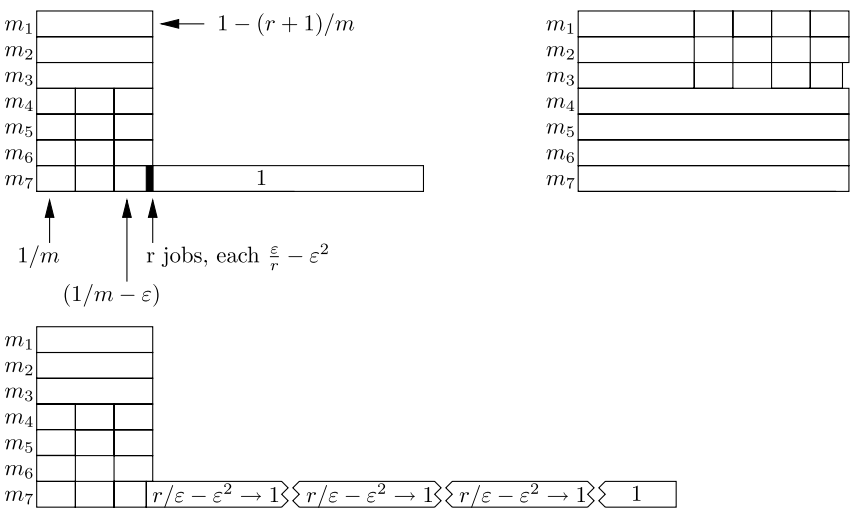

Fig. 3 A worst-case instance for arbitrary increase of r jobs. *Top left*, Graham's schedule before the perturbation. *Bottom left*, Graham's schedule after the increase. *Top right*, the optimal perturbed schedule

$= (r+1)(m-r-1)-1$ small jobs with processing time $\frac{1}{m}$ next, followed by one small job with processing time $\frac{1}{m}-\varepsilon$, for ε arbitrarily small. Graham's algorithm schedules the jobs evenly on the machines processing no big jobs. At this point, all but one machine have a load of $1-\frac{r+1}{m}$, and one machine has load $1-\frac{r+1}{m}-\varepsilon$. Now, the adversary presents r tiny jobs with processing time $\frac{\varepsilon}{r}-\varepsilon^2$ and a last job with processing time 1. The perturbation increases the processing times of tiny jobs from $\frac{\varepsilon}{r}-\varepsilon^2$ to 1. The makespan of Graham's schedule is $2+r-\frac{r+1}{m}-\varepsilon$. On the other hand, the optimal offline algorithm on the perturbed instance schedules each of the now m big jobs on a different machine, and then schedules $r+1$ small jobs on each machine processing a big job with processing time $1-\frac{r+1}{m}$, thus achieving the makespan of 1. □

Theorem 3 can be generalized as follows. Let $J_i^{\uparrow}$ be the set of perturbed jobs which are scheduled on machine $i \in M$ in an optimum offline solution $\mathrm{OPT}(J, P^{\uparrow})$. Let $M^{\neq} = \{i \in M | J_i^{\uparrow} \neq \emptyset\}$ be the set of machines in the considered optimal offline solution which process at least one job with perturbed processing time, and define $r = |M^{\neq}|$. Note that here the perturbed machines are defined with respect to $\mathrm{OPT}(J, P^{\uparrow})$, and not to Graham's schedule.

Theorem 4 *Consider the instances of online scheduling on m identical parallel machines with the following properties*: *first*, *the processing times of some jobs may increase arbitrarily*; *second*, *the perturbed jobs are scheduled on r machines of an optimal offline perturbed solution*. *For these instances*, *Graham's algorithm has a competitive ratio of* $2+r-\frac{r+1}{m}$, *and this bound is best possible for* $r \leq m-2$.

Proof Let $J^{\uparrow} = J_1^{\uparrow} \cup J_2^{\uparrow} \cup \cdots \cup J_m^{\uparrow}$ be the union of the sets $J_i^{\uparrow}$. Let $\psi_j = p_j^{\uparrow} - p_j$, $j \in J^{\uparrow}$ be the increase in processing time of the perturbed jobs. For the analysis of Graham's schedule, we may assume that the increases in processing time of the jobs in $J^{\uparrow}$ occur sequentially in an arbitrary order. Therefore, each job $j \in J^{\uparrow}$ causes a well defined variation δ_j^k in idle time on each machine $k \in M$ of Graham's schedule. This variation is bounded by $-\psi_j \leq \delta_j^k \leq \psi_j$; furthermore, this variation is negative or zero on the machine where $j \in J^{\uparrow}$ is scheduled by Graham's algorithm. Thus, $\sum_{k \in M} \delta_j^k \leq (m-1) \cdot \psi_j$, $j \in J^{\uparrow}$. Finally, $\sum_{j \in J_i^{\uparrow}} \psi_j \leq \sum_{j \in J_i^{\uparrow}} p_j^{\uparrow} \leq \mathcal{L}_{\mathrm{OPT}}^{\uparrow}$, $i \in M$, since *all* the jobs in $J_i^{\uparrow}$ are scheduled on the same machine in the considered optimal offline schedule of the perturbed instance. Therefore,

$$m \cdot \mathcal{L}_{\mathrm{List}}^{\uparrow} = \sum_{j \in J} p_j + \sum_{i \in M} s_i + \sum_{j \in J^{\uparrow}} \psi_j + \sum_{j \in J^{\uparrow}} \sum_{k \in M} \delta_j^k,$$

and we can bound the increases in idle time as follows:

$$\sum_{j \in J^{\uparrow}} \sum_{k \in M} \delta_j^k = \sum_{i \in M^{\neq}} \sum_{j \in J_i^{\uparrow}} \sum_{k \in M} \delta_j^k \leq \sum_{i \in M^{\neq}} \sum_{j \in J_i^{\uparrow}} (m-1)\psi_j \leq (m-1) \sum_{i \in M^{\neq}} \mathcal{L}_{\mathrm{OPT}}^{\uparrow} \leq (m-1) \cdot r \cdot \mathcal{L}_{\mathrm{OPT}}^{\uparrow}.$$

Hence, we have

$$\begin{aligned} m \cdot \mathcal{L}_{\mathrm{List}}^{\uparrow} &\leq \sum_{j \in J} p_j + \sum_{i \in M} s_i + \sum_{j \in J^{\uparrow}} \psi_j + r(m-1)\mathcal{L}_{\mathrm{OPT}}^{\uparrow} \\ &= \sum_{j \in J} p_j^{\uparrow} + \sum_{i \in M} s_i + r(m-1)\mathcal{L}_{\mathrm{OPT}}^{\uparrow} \\ &\leq m \cdot \mathcal{L}_{\mathrm{OPT}}^{\uparrow} + (m-1) \cdot \mathcal{L}_{\mathrm{OPT}}^{\uparrow} + r(m-1)\mathcal{L}_{\mathrm{OPT}}^{\uparrow}, \end{aligned}$$

since at least one machine attained the makespan before the perturbation and had zero idle time. Thus, we have the following bound on the makespan:

$$\mathcal{L}\big(\mathrm{List}(J, P), P^{\uparrow}\big) \leq \left(2 + r - \frac{r+1}{m}\right) \times \mathcal{L}\big(\mathrm{OPT}\big(J, P^{\uparrow}\big), P^{\uparrow}\big).$$

The bound is best-possible for $r \leq m-2$, since the example of Theorem 3 affects r machines. □

4 Bounded perturbations

In the following analyses, we constrain the effect perturbations may have. Here, we assume that the perturbed processing time is related to the original processing time, and bound the perturbed processing times by a constant factor of their original processing time. Thus, the perturbed processing times are of the form $\tilde{p} = \alpha p$ for some α. Values of $\alpha > 1$ imply perturbations increasing the processing times, whereas $\alpha < 1$ imply perturbations decreasing processing times. In this section, we analyze both cases separately.

4.1 Bounded decreases in processing times

In this section, we analyze the behavior of the makespan given that the processing times of jobs may decrease by a bounded factor. We start by stating a simple lower bound.

Theorem 5 *Consider the instances of online scheduling on $m \geq 2$ identical parallel machines where the processing time of at most one job may decrease to a factor at least $\frac{1}{x}$ of its original processing time*, *for $x > 1$, $x \in \mathbb{Q}_0^+$. No online algorithm applied to these instances can achieve a competitive ratio smaller than $2 - \frac{2}{x+1}$.*

The stated bound is $\frac{4}{3}$ for halving processing times, and is arbitrarily close to the lower bound of 2 stated in Theorem 1 for arbitrarily large x.

Proof Consider an adversarial sequence of $m+1$ jobs of processing time 1. Any algorithm aiming at a competitive ratio strictly smaller than 2 must schedule the first m jobs each on a different machine. Were this not so, the adversary could stop the sequence after the first job has been scheduled on a machine with nonzero load, thus enforcing a competitive ratio of two. The last job may be scheduled on any machine. Now, the perturbation decreases the processing time of a job which is scheduled alone on a machine from 1 to $\frac{1}{x}$. Any online algorithm aiming at a competitive ratio smaller than two has a makespan of two, whereas the optimum offline perturbed schedule achieves a makespan of $1+\frac{1}{x}$ by scheduling a job of processing time 1 on the machine where the perturbed job is scheduled, and by scheduling the remaining $m-1$ jobs each on one machine. The lower bound follows from the ratio of the two makespans. □

A lower bound of 2 on the competitive ratio can be shown for the case where at most 2 jobs at most halve their processing time due to the perturbations:

Theorem 6 *Consider the instances of online scheduling on $m \geq 3$ identical parallel machines where the processing time of at most two jobs may decrease to a factor at least $\frac{1}{x}$ of their original processing time, with $x > 1, x \in \mathbb{Q}_0^+$. Applied to these instances, no online algorithm can have a competitive ratio strictly smaller than* $\min\{x, 2\}$.

Proof By contradiction. Assume an online algorithms with a strictly smaller competitive ratio exists, and consider an adversarial sequence of $m+1$ jobs of processing time 1. By the same argument as in the proof of Theorem 5, any online algorithm aiming at a competitive ratio strictly smaller than 2 has a makespan of two. The perturbation decreases the processing times of two jobs that are scheduled alone each on one machine to $\frac{1}{x}$. This perturbation leaves the makespan of the schedule computed by the online algorithm unchanged. The optimum offline perturbed schedule achieves a makespan of $\max\{1, 2\cdot\frac{1}{x}\}$ by scheduling the two perturbed jobs on the same machine and the remaining $m-1$ jobs each on a machine. For $x \in [1,2]$ the ratio of the two objectives is x, and for $x > 2$ the ratio is two, a contradiction to our assumption. □

Having assessed these lower bounds, we analyze the quality of the schedules built by Graham's algorithm when facing these kinds of perturbations.

Theorem 7 *Consider the instances of online scheduling on m identical parallel machines where the processing times of an arbitrary number of jobs may decrease to a factor at least $\frac{1}{x}$ of their original processing time, for $x > 1, x \in \mathbb{Q}_0^+$. Restricted to these instances, Graham's algorithm has a competitive ratio between $1+x-\frac{x^2}{m-1+x}-\varepsilon$ and $1+x-\frac{x}{m}$, for a small $\varepsilon \in \mathbb{Q}_0^+, 0 < \varepsilon < \frac{x}{m-1}(1-\frac{x}{m-1+x})$.*

Proof Consider a machine μ attaining the makespan after the perturbation. Let $\bar{j}_\mu$ be the last job scheduled on μ, and let ℓ be the load of μ on the original instance when $\bar{j}_\mu$ was presented. Hence, the total load of μ before the perturbation is $\ell + p_{\bar{j}_\mu}$. Because $\bar{j}_\mu$ was scheduled on μ, all machines have at least load ℓ with the original processing times. Therefore, $m\ell \leq \sum_{j\in J\setminus\{\bar{j}_\mu\}} p_j$. Since at most all jobs are perturbed and decrease to a factor of at least $\frac{1}{x}$, $\sum_{j\in J\setminus\{\bar{j}_\mu\}} p_j \leq x\sum_{j\in J\setminus\{\bar{j}_\mu\}} p_j^\downarrow \leq xm\cdot\mathcal{L}_{\mathrm{OPT}}^\downarrow - x\cdot p_{\bar{j}_\mu}^\downarrow$, hence $\ell \leq x\mathcal{L}_{\mathrm{OPT}}^\downarrow - \frac{x}{m}p_{\bar{j}_\mu}^\downarrow$. Thus,

$$\begin{aligned}\mathcal{L}\big(\mathrm{List}(J,P),P^\downarrow\big) &\leq \ell + p_{\bar{j}_\mu}^\downarrow \leq x\cdot\mathcal{L}_{\mathrm{OPT}}^\downarrow - \frac{x}{m}p_{\bar{j}_\mu}^\downarrow + p_{\bar{j}_\mu}^\downarrow\\ &\leq x\cdot\mathcal{L}_{\mathrm{OPT}}^\downarrow + \left(1-\frac{x}{m}\right)p_{\bar{j}_\mu}^\downarrow\\ &\leq \left(1+x-\frac{x}{m}\right)\mathcal{L}\big(\mathrm{OPT}\big(J,P^\downarrow\big),P^\downarrow\big).\end{aligned}$$

An example which comes close to this upper bound is the following sequence of jobs, illustrated in Fig. 4 for the case $x=2$. The adversary presents $m-1$ big jobs with processing time $x-\frac{x^2}{m-1+x}$, $m-2$ small jobs with processing time $\frac{x}{m-1}(1-\frac{x}{m-1+x})$, one small job with processing time $\frac{x}{m-1}(1-\frac{x}{m-1+x})-\varepsilon$, for a small $\varepsilon\in\mathbb{Q}_0^+, 0<\varepsilon<\frac{x}{m-1}(1-\frac{x}{m-1+x})$, followed by the last job with processing time 1. The perturbation affects the $m-1$ big jobs, decreasing their processing time to $1-\frac{x}{m-1+x}$. Graham's schedule on the perturbed instance has a makespan of $1+x-\frac{x^2}{m-1+x}-\varepsilon$, whereas the optimum offline perturbed solution has a makespan $\mathcal{L}(\mathrm{OPT}(J,P^\downarrow),P^\downarrow)=1$. □

Corollary 8 *Consider the instances of online scheduling on m identical parallel machines where the processing times of an arbitrary number of jobs may halve. Restricted to these instances, Graham's algorithm has a competitive ratio of at most $3-\frac{2}{m}$, and at least $3-\frac{4}{m+1}-\varepsilon$, for $\varepsilon\in\mathbb{Q}_0^+, 0<\varepsilon<\frac{2}{m-1}-\frac{4}{m^2-1}, m>1$.*

The corollary follows by setting $x=2$ in Theorem 7. By setting $x=1$, we match the bound for Graham's algorithm.

In the following, we bound the number of jobs which a perturbation may decrease. We show the following theorem:

Theorem 9 *Consider the instances of online scheduling on m identical parallel machines where the perturbations may decrease the processing times of r jobs to a factor of at least*

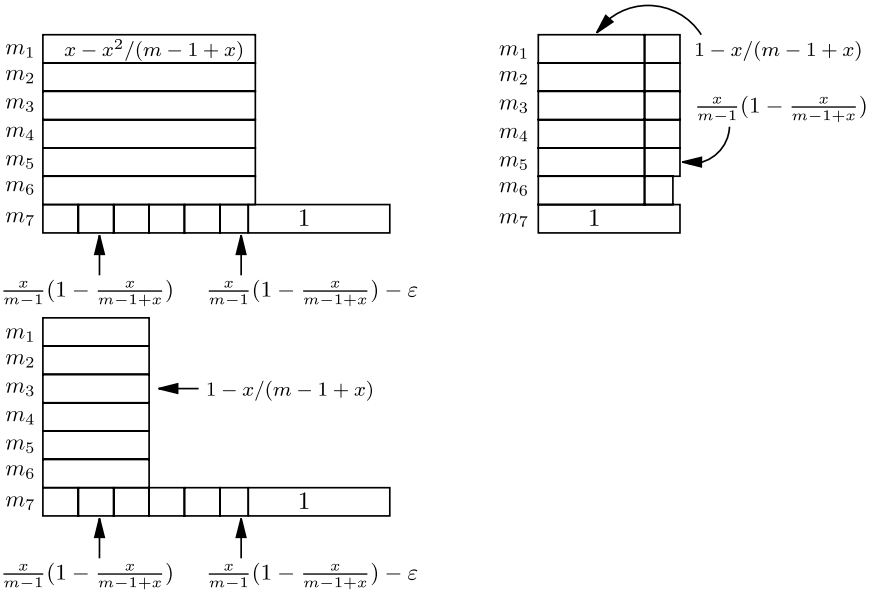

Fig. 4 The bad example of Theorem 7 for $x = 2$. *Top left*, the Graham's schedule on the original instance. *Bottom left*, the Graham's schedule after the perturbation. *Right*, the optimum offline perturbed schedule

$\frac{1}{x}$ of their original processing time, for $x \in \mathbb{Q}_0^+, x > 1$. Restricted to these instances, Graham's algorithm has a competitive ratio of $2 + \frac{r\cdot(x-1)-1}{m}$.

Proof Consider a machine μ attaining the makespan after the perturbation. Let $\bar{j}_\mu$ be the last job scheduled on μ, and let ℓ be the load of μ in the original instance when $\bar{j}_\mu$ was presented. Therefore, $m\ell \le \sum_{j\in J\setminus\{\bar{j}_\mu\}} p_j$. Let J_r be the set of the r perturbed jobs with the exception of $\bar{j}_\mu$, given that $\bar{j}_\mu$ was perturbed. Then, we have:

$$\begin{aligned} m\ell &\le \sum_{j\in J\setminus\{\bar{j}_\mu\}\setminus J_r} p_j + \sum_{j\in J_r} p_j \\ &\le \sum_{j\in J\setminus\{\bar{j}_\mu\}\setminus J_r} p_j^{\downarrow} + x\cdot\sum_{j\in J_r} p_j^{\downarrow} \\ &= \sum_{j\in J} p_j^{\downarrow} + (x-1)\sum_{j\in J_r} p_j^{\downarrow} - p_{\bar{j}_\mu}^{\downarrow} \\ &\le \big(m + (x-1)\cdot r\big)\mathcal{L}_{\mathrm{OPT}}^{\downarrow} - p_{\bar{j}_\mu}^{\downarrow}, \end{aligned}$$

since $p_j^{\downarrow} \le \mathcal{L}_{\mathrm{OPT}}^{\downarrow}$, $j \in J_r$. Finally, as the makespan of Graham's algorithm can be bounded by $\mathcal{L}_{\mathrm{List}}^{\downarrow} \le \ell + p_{\bar{j}_\mu}^{\downarrow}$, we have:

$$\mathcal{L}\big(\mathrm{List}(J,P),P^{\downarrow}\big) \le \left(2 + \frac{(x-1)\cdot r - 1}{m}\right) \times \mathcal{L}\big(\mathrm{OPT}\big(J,P^{\downarrow}\big),P^{\downarrow}\big). \qquad \square$$

Corollary 10 *Graham's algorithm for online scheduling on m identical parallel machines, applied to instances where the processing time of one job may halve, has a competitive ratio not greater than 2 and no smaller than $2 - \frac{1}{2m-1} - \varepsilon$, for an arbitrary small $\varepsilon \in \mathbb{Q}_0^+, 0 < \varepsilon < \frac{1}{2m-1}$.*

Proof For the upper bound, it is sufficient to set $x = 2$ and $r = 1$ in Theorem 9. A bad example achieving the lower bound is the following sequence of jobs: the adversary presents $m-1$ big jobs with processing time $1 - \frac{1}{2m-1}$, which Graham's algorithm schedules each on a different machine. Then, $2(m-1) - 1$ small jobs with processing time $\frac{1}{2m-1}$ and a small job with processing time $\frac{1}{2m-1} - \varepsilon$, $\varepsilon \in \mathbb{Q}_0^+, 0 < \varepsilon < \frac{1}{2m-1}$, follow. All small jobs are scheduled on the same initially empty machine. Finally, the adversary presents a job with processing time 1, which is scheduled together with the small jobs. Now, the perturbation affects any one of the $(m-1)$ big jobs, decreasing its processing time to $\frac{1}{2} - \frac{1}{2}\frac{1}{2m-1}$. The makespan of this schedule is $2 - \frac{1}{2m-1} - \varepsilon$. The optimum offline perturbed scheduled has a makespan of 1: it schedules each of the $m-2$ jobs with processing time $1 - \frac{1}{2m-1}$ together with one job of processing time $\frac{1}{2m-1}$ on a separate machine, the job of processing time 1 alone on one machine, and the remaining m small jobs with the job of processing time $\frac{1}{2} - \frac{1}{2}\frac{1}{2m-1}$ on the remaining machine. $\square$

Similar to Theorem 4, Theorem 7 can also be stated with respect to the number r of affected machines $M^{\neq}$ in an optimum offline solution $\mathrm{OPT}(J, P^{\downarrow})$.

Theorem 11 *Consider the instances for online scheduling on m identical parallel machines with the following two properties: first, the perturbations may decrease the processing times of some jobs to a factor of at least $\frac{1}{x}$ of their original processing time, for $x \in \mathbb{Q}_0^+, x > 1$; second, the perturbed jobs are scheduled on r machines in an optimum of-*

fline solution. Restricted to these instances, Graham's algorithm has a competitive ratio of $2 + \frac{r\cdot(x-1)-1}{m}$.

Proof Let $J_i^{\downarrow}$ be the set of perturbed jobs scheduled on machine $i \in M$ in $\mathrm{OPT}(J, P^{\downarrow})$, and let $M^{\neq} = \{i \in M | J_i^{\downarrow} \neq \emptyset\}$, $r = |M^{\neq}|$. Let $\tilde{J} = \bigcup_{i\in M^{\neq}} J_i^{\downarrow}$ be the set of all perturbed jobs. Consider a machine μ attaining the makespan after the perturbation, let $\bar{j}_\mu$ be the last job scheduled on μ, and let ℓ be the load of μ in the original instance when $\bar{j}_\mu$ was presented. Therefore, $\mathcal{L}(\mathrm{List}(J, P), P^{\downarrow}) \leq \ell + p_{\bar{j}_\mu}$. Now:

$$\begin{aligned}
m\ell &\leq \sum_{j\in J\setminus\{\bar{j}_\mu\}} p_j \leq \sum_{j\in\tilde{J}} p_j + \sum_{j\in J\setminus\tilde{J}} p_j - p_{\bar{j}_\mu} \\
&\leq x\sum_{j\in\tilde{J}} p_j^{\downarrow} + \sum_{j\in J\setminus\tilde{J}} p_j^{\downarrow} - p_{\bar{j}_\mu} \\
&\leq \sum_{j\in J} p_j^{\downarrow} + (x-1)\sum_{j\in\tilde{J}} p_j^{\downarrow} - p_{\bar{j}_\mu} \\
&\leq \sum_{j\in J} p_j^{\downarrow} + (x-1)\sum_{i\in M^{\neq}}\sum_{j\in J_i^{\downarrow}} p_j^{\downarrow} - p_{\bar{j}_\mu} \\
&\leq m\mathcal{L}_{\mathrm{OPT}}^{\downarrow} + (x-1)r\mathcal{L}_{\mathrm{OPT}}^{\downarrow} - p_{\bar{j}_\mu} \\
&= \mathcal{L}_{\mathrm{OPT}}^{\downarrow}\big(m + (x-1)r\big) - p_{\bar{j}_\mu},
\end{aligned}$$

where the last inequality follows because $\sum_{j\in J_i^{\downarrow}} p_j^{\downarrow} \leq \mathcal{L}_{\mathrm{OPT}}^{\downarrow}$. Thus,

$$\begin{aligned}
\mathcal{L}_{\mathrm{List}}^{\downarrow} &\leq \mathcal{L}_{\mathrm{OPT}}^{\downarrow} + \frac{(x-1)r}{m}\mathcal{L}_{\mathrm{OPT}}^{\downarrow} + \left(1 - \frac{1}{m}\right)p_{\bar{j}_\mu} \\
&\leq \left(2 + \frac{(x-1)r-1}{m}\right)\mathcal{L}_{\mathrm{OPT}}^{\downarrow}.
\end{aligned}$$

□

4.2 Bounded increases in processing times

We start by analyzing the impact of doubling the processing time of one job, proceed by considering the increase by a constant factor of one job's processing time, then extend the analysis to allowing many jobs to change in processing time by a bounded amount.

Surprisingly, we shall see that the last analysis suggests that it is not that relevant how many jobs we are allowed to increase in processing time, but that if we allow an x-factor increase in processing time, x perturbed jobs are sufficient to produce a worst-case behavior.

4.2.1 Bounded increase of one job

In this section, we first show a simple lower bound for any online algorithm on instances where the processing time of one job may double; then, we show that this setting does not affect Graham's algorithm as badly as the arbitrary increase of one job.

Theorem 12 *No algorithm for online scheduling on $m \geq 2$ identical parallel machines can have a competitive ratio strictly smaller than $\frac{3}{2}$ if the processing time of one job may double.*

Proof Assume this was possible and consider an online algorithm which achieves a competitive ratio better than $\frac{3}{2}$. The adversarial sequence consists of $m + 1$ jobs with processing time 1. To achieve a competitive ratio smaller than 2, the online algorithm schedules the first m jobs each on a different machine. The last job may be scheduled on any machine. The perturbation now doubles the processing time of the job scheduled last. By doing so, this schedule has a makespan of 3, whereas the optimal offline perturbed schedule has a makespan of 2, obtained by scheduling the only job with processing time 2 alone on a machine, and by scheduling the other jobs in such a way that the maximum load of any machine is 2. □

We now analyze the performance of Graham's algorithm in this situation, but for general bounded increase. The analysis shows a swift transition from the case of bounded perturbation to arbitrary perturbation of one job.

Theorem 13 *Graham's algorithm for online scheduling on m identical parallel machines on instances where the processing time of one job is perturbed to a factor $x > 1$, $x \in \mathbb{Q}_0^+$ of its original processing time has a competitive ratio of $3 - \frac{1}{x} - \frac{(2-\frac{1}{x})}{m}$. This bound is best possible for $m \geq 3$ and $x \geq \frac{m-1}{m-2}$.*

Proof Let $\tilde{j}$ be the perturbed job. In this case, $p_{\tilde{j}}^{\uparrow} = x \cdot p_{\tilde{j}}$. Therefore, the idle time of each machine may increase by at most the amount of this increase, that is, by $\frac{x-1}{x}p_{\tilde{j}}^{\uparrow}$, and the machine μ where $\tilde{j}$ is scheduled has no increase. Naturally, since the optimal offline schedule needs to process $\tilde{j}$ as well, $p_{\tilde{j}}^{\uparrow} \leq \mathcal{L}_{\mathrm{OPT}}^{\uparrow}$. Finally, since the processing times may only increase, the idle times of the original schedule are bounded by $\mathcal{L}_{\mathrm{OPT}}^{\uparrow}$, and the machine $\bar{\mu}$ which attains the makespan $\mathcal{L}(\mathrm{List}(J, P), P)$ has zero idle time. Now, the schedule of the machines up to the makespan $\mathcal{L}(\mathrm{List}(J, P), P^{\uparrow})$ is described as follows:

$$\begin{aligned}
m \cdot \mathcal{L}_{\mathrm{List}}^{\uparrow} &= \sum_{j\in J} p_j^{\uparrow} + \sum_{i\in M} s_i^{\uparrow} \\
&\leq \sum_{j\in J} p_j^{\uparrow} + \sum_{i\in M\setminus\{\bar{\mu}\}} s_i + \sum_{i\in M\setminus\{\mu\}} \frac{x-1}{x}p_{\tilde{j}}^{\uparrow} \\
&\leq m \cdot \mathcal{L}_{\mathrm{OPT}}^{\uparrow} + (m-1)\cdot\left(\mathcal{L}_{\mathrm{OPT}}^{\uparrow} + \frac{x-1}{x}\mathcal{L}_{\mathrm{OPT}}^{\uparrow}\right).
\end{aligned}$$

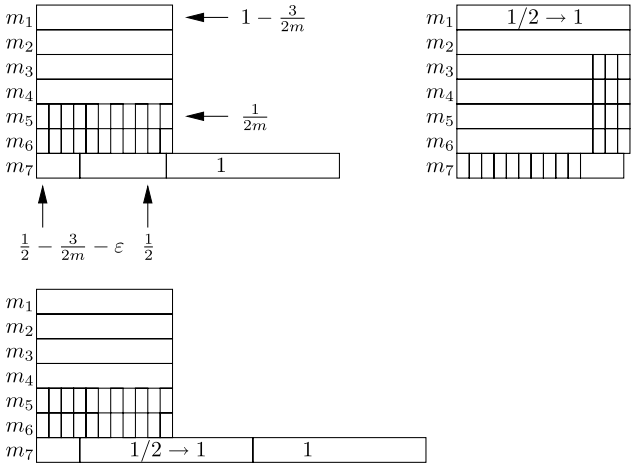

Fig. 5 A worst-case example for doubling the processing time of one job. *Top left*, Graham's schedule before the increase. *Bottom left*, Graham's schedule after the increase. *Top*, the optimum offline perturbed schedule

Thus,

$$\mathcal{L}\big(\mathrm{List}(J,P),P^{\uparrow}\big) \le \left(3-\frac{1}{x}-\frac{(2-\frac{1}{x})}{m}\right) \times \mathcal{L}\big(\mathrm{OPT}\big(J,P^{\uparrow}\big),P^{\uparrow}\big).$$

An example coming arbitrarily close to this bound for $x = \frac{p}{q} \in \mathbb{Q}_0^+$ is as follows, and shown in Fig. 5 for the case $x = \frac{2}{1}$. Note that the restriction $m \ge 3$ is necessary since at least 3 machines are needed to schedule the jobs with a makespan of 1 for the optimum offline perturbed schedule, and $x \ge \frac{m-1}{m-2}$ is required for ensuring that all processing times are positive. The adversary sequentially presents $m-3$ jobs with processing time $1-\frac{(2-\frac{1}{x})}{m}$, one job with processing time $1-\frac{1}{x}-\frac{2-\frac{1}{x}}{m}-\varepsilon$, $2pm-2qm-4p+2q$ jobs with processing time $\frac{1}{pm}$, one job with processing time $\frac{1}{x}$, $2qm$ jobs with processing time $\frac{1}{pm}$ and finally a job with processing time 1. The perturbation increases the job with processing time $\frac{1}{x}$ to 1, leading to a makespan of $(3-\frac{1}{x}-\frac{(2-\frac{1}{x})}{m})$, whereas the optimum offline algorithm on the perturbed instance achieves a makespan of 1. □

We remark that for big x we get arbitrarily close to the result of Theorem 3 for $r = 1$. Finally, with $x = 1$ we match the bound of Graham's algorithm.

4.2.2 Bounded increase of many jobs

We again begin by showing a simple lower bound.

Theorem 14 *No online algorithm for online scheduling on m identical parallel machines can have a competitive ratio strictly smaller than* 2 *if the processing time of two jobs may double.*

Proof The proof is similar to the proof of Theorem 12. Assume an online algorithm with a competitive ratio strictly smaller than 2 exists. The adversarial sequence for this algorithm consists of $m+1$ jobs of processing time 1. By the same arguments of Theorem 5, no two jobs within the first m can be scheduled on the same machine by the online algorithm. The last of the jobs of the sequence can be scheduled on any machine. Consider the jobs scheduled on the machine having a load of 2. The worst-case perturbation doubles the processing time of these jobs, enforcing a makespan of 4. The optimum offline perturbed schedule, on the other hand, schedules the two jobs of processing time 2 each on a single machine, and distributes the remaining $m-1$ jobs of processing time 1 on $m-2$ machines in such a way that no machine exceeds a load of 2. The ratio of the two makespans proves a competitive ratio of 2, a contradiction. □

The previous analyses of Graham's algorithm can be extended to the cases where the perturbation affects more than one job. Our analysis shows that the actual number of perturbed jobs is not really relevant, but is tied to the amount of the perturbation.

Theorem 15 *Consider the instances of online scheduling on m identical parallel machines where perturbations may increase the processing times of many jobs, each to a factor at most* $x \ge 1$, $x \in \mathbb{Q}_0^+$ *of their original processing time. Restricted to these instances, Graham's algorithm is* $1+x-\frac{1}{m}$*-competitive. For* $x \in \mathbb{N}$ *and* $x \le m-1$, *the competitive ratio of Graham's algorithm is at least* $1+x-\frac{x^2}{m-1+x}-\varepsilon$, *for arbitrarily small* $\varepsilon \in \mathbb{Q}_0^+$, $0 < \varepsilon < \frac{1}{x\cdot(m-1+x)}$.

Proof Consider a machine μ attaining the makespan $\mathcal{L}_{\mathrm{List}}^{\uparrow}$ as it was in the original instance. We partition the processing times of the jobs on μ as follows. Let $p_{\bar{\jmath}}$ be the processing time of the last job $\bar{\jmath}$ scheduled on μ, β be the processing time of perturbed jobs excluding job $\bar{\jmath}$ if it is perturbed, and γ be the processing time of unperturbed jobs excluding job $\bar{\jmath}$ if it is unperturbed. Due to Graham's algorithm, $\beta+\gamma \le \mathcal{L}_{\mathrm{OPT}} - \frac{p_{\bar{\jmath}}}{m}$, since, on the unperturbed instance, when the last job $\bar{\jmath}$ was scheduled on μ all machines had at least this load. Thus, $\beta \le \mathcal{L}_{\mathrm{OPT}} - \frac{p_{\bar{\jmath}}}{m} - \gamma$. Because the processing times can only increase, $\mathcal{L}_{\mathrm{OPT}} \le \mathcal{L}_{\mathrm{OPT}}^{\uparrow}$. Finally, let $p_{\bar{\jmath}}^{\uparrow}$ and $\beta^{\uparrow}$ be the perturbed counterpart of the processing times $p_{\bar{\jmath}}$ and β. Let $z \in [1,x]$ be the actual increase factor of job $\bar{\jmath}$; thus, $p_{\bar{\jmath}}^{\uparrow} = z p_{\bar{\jmath}} \le x p_{\bar{\jmath}}$. The makespan $\mathcal{L}(\mathrm{List}(J,P),P^{\uparrow})$ can be bounded as follows:

$$\begin{aligned}\mathcal{L}_{\mathrm{List}}^{\uparrow} &= \gamma + \beta^{\uparrow} + p_{\bar{\jmath}}^{\uparrow} \\ &\le \gamma + x\cdot\beta + p_{\bar{\jmath}}^{\uparrow} \\ &\le \gamma + x\left(\mathcal{L}_{\mathrm{OPT}} - \gamma - \frac{p_{\bar{\jmath}}}{m}\right) + p_{\bar{\jmath}}^{\uparrow}\end{aligned}$$

$$\leq (1-x)\gamma + x\mathcal{L}^{\uparrow}_{\mathrm{OPT}} + p^{\uparrow}_{\bar{j}} - \frac{xp_{\bar{j}}}{m}$$
$$\leq x\mathcal{L}^{\uparrow}_{\mathrm{OPT}} + p^{\uparrow}_{\bar{j}} - \frac{zp_{\bar{j}}}{m}$$
$$\leq x\mathcal{L}^{\uparrow}_{\mathrm{OPT}} + p^{\uparrow}_{\bar{j}} - \frac{p^{\uparrow}_{\bar{j}}}{m} \leq \left(1 + x - \frac{1}{m}\right)\mathcal{L}^{\uparrow}_{\mathrm{OPT}}.$$

The fourth inequality holds because of $zp_{\bar{j}} \leq xp_{\bar{j}}$ and $1 - x \leq 0$, the fifth one is satisfied with equality because of $p^{\uparrow}_{\bar{j}} = zp_{\bar{j}}$, and the last inequality because of $p^{\uparrow}_{\bar{j}} \leq \mathcal{L}^{\uparrow}_{\mathrm{OPT}}$. Therefore,

$$\mathcal{L}\big(\mathrm{List}(J,P), P^{\uparrow}\big) \leq \left(1 + x - \frac{1}{m}\right)\mathcal{L}\big(\mathrm{OPT}\big(J, P^{\uparrow}\big), P^{\uparrow}\big).$$

An example achieving a bad competitive ratio for $x \leq m-1, x \in \mathbb{N}$, has the following structure, shown in Fig. 6. First, the adversary presents $m-x-1$ big jobs with processing time $1 - \frac{x}{m-1+x}$. The online algorithm schedules them on idle machines. Next, the adversary presents the following sequence of jobs: one medium job with processing time $\frac{1}{x} - \frac{1}{m-1+x} - \varepsilon$, with $\varepsilon \in \mathbb{Q}^{+}_{0}, 0 < \varepsilon < \frac{1}{x\cdot(m-1+x)}$, and $(m-1)\cdot x$ small jobs with processing time $\frac{1}{x\cdot(m-1+x)}$. The same sequence is repeated $x-1$ times with the difference that the medium jobs have processing time $\frac{1}{x} - \frac{1}{m-1+x}$. Because of the offset ε of the first medium job, all medium jobs are scheduled on the same machine, and the small jobs are distributed evenly on x machines. At last, the adversary presents a big job with processing time 1, which is scheduled on the machine processing the medium jobs. Now, the perturbation affects the medium jobs, which increase to $1 - \frac{x}{m-1+x}$ (except for the first medium job, which is εx smaller than that) and enforce a makespan of $1 + x - \frac{x^2}{m-1+x} - x\varepsilon$. The optimal offline algorithm, on the other hand, achieves a makespan of 1 by scheduling each big job on a different machine, and by scheduling x^2 small jobs with processing time $\frac{1}{x\cdot(m-1+x)}$ on each of the $m-1$ machines which do not yet attain the load of 1 with their big job. Note that in this example it is sufficient to perturb x jobs to get this bad behavior. □

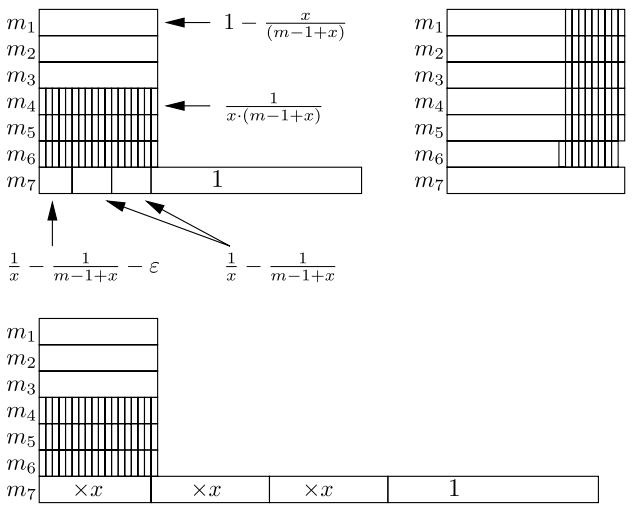

Fig. 6 The example of Theorem 15, here with $x = 3$. *Top left*, the schedule produced by Graham's algorithm with the original instance. *Bottom left*, the schedule after the perturbations. *Top right*, the optimum offline perturbed schedule

In the following, we consider increases in processing times which may be different for all jobs, but where the impact on the schedule is bounded. To that end, consider Graham's schedule $\mathrm{List}(J,P)$. For each machine $i \in M$, we partition its assigned jobs as follows: Let $\bar{j}_i$ be the last job scheduled on machine i, P_i be the set of jobs, excluding $\bar{j}_i$, which are not perturbed, and $\tilde{P}_i$ the set of jobs, excluding $\bar{j}_i$, which are perturbed.

Theorem 16 *Consider the instances of online scheduling on m identical parallel machines where the processing times of many jobs may increase. Assume that a machine μ attaining the makespan in a schedule obtained by Graham's algorithm has the following properties*: $\sum_{j\in\tilde{P}_\mu} p^{\uparrow}_{j} = x\sum_{j\in\tilde{P}_\mu} p_j$ *and* $p^{\uparrow}_{\bar{j}_\mu} = yp_{\bar{j}_\mu}$. *In this case, Graham's algorithm has a competitive ratio of* $1 + x - \frac{x}{ym}$, *for* $x \geq 1$, $x \in \mathbb{Q}^{+}_{0}, y \geq 1, y \in \mathbb{Q}^{+}_{0}$.

Proof Consider a machine μ attaining the makespan $\mathcal{L}^{\uparrow}_{\mathrm{List}}$ in its unperturbed state. When Graham's algorithm considered $\bar{j}_\mu$, each machine had a load of at least $\sum_{j\in\tilde{P}_\mu} p_j + \sum_{j\in P_\mu} p_j \leq \mathcal{L}_{\mathrm{OPT}} - \frac{p_{\bar{j}_\mu}}{m}$. Thus, $\sum_{j\in\tilde{P}_\mu} p_j \leq \mathcal{L}_{\mathrm{OPT}} - \frac{p_{\bar{j}_\mu}}{m} - \sum_{j\in P_\mu} p_j$. Because the processing times increase, $\mathcal{L}_{\mathrm{OPT}} \leq \mathcal{L}^{\uparrow}_{\mathrm{OPT}}$. Finally, $yp_{\bar{j}_\mu} \leq \mathcal{L}^{\uparrow}_{\mathrm{OPT}}$, since $\bar{j}_\mu$ must also be scheduled. The makespan $\mathcal{L}(\mathrm{List}(J,P), P^{\uparrow})$ can be described as follows:

$$\mathcal{L}^{\uparrow}_{\mathrm{List}} = \sum_{j\in P_\mu} p_j + x\sum_{j\in\tilde{P}_\mu} p_j + yp_{\bar{j}_\mu}$$
$$\leq \sum_{j\in P_\mu} p_j + x\left(\mathcal{L}_{\mathrm{OPT}} - \frac{p_{\bar{j}_\mu}}{m} - \sum_{j\in P_\mu} p_j\right) + yp_{\bar{j}_\mu}$$
$$\leq \sum_{j\in P_\mu} p_j + x\mathcal{L}_{\mathrm{OPT}} - x\frac{p_{\bar{j}_\mu}}{m} - x\sum_{j\in P_\mu} p_j + yp_{\bar{j}_\mu}$$
$$\leq (1-x)\sum_{j\in P_\mu} p_j + x\mathcal{L}_{\mathrm{OPT}} + yp_{\bar{j}_\mu}\left(1 - \frac{x}{ym}\right)$$
$$\leq \left(1 + x - \frac{x}{ym}\right)\mathcal{L}^{\uparrow}_{\mathrm{OPT}}.$$

Thus,

$$\mathcal{L}\big(\mathrm{List}(J,P), P^{\uparrow}\big) \leq \left(1 + x - \frac{x}{ym}\right)\mathcal{L}\big(\mathrm{OPT}\big(J, P^{\uparrow}\big), P^{\uparrow}\big).$$

□

5 A Graham-like risk aversion algorithm

In an effort to keep the impact of perturbations under control, the following strategy could prove to be effective for online scheduling. When considering the next job in the sequence, one could enumerate all possible assignments of this job to the machines. For each such assignment, we could evaluate the effect on the competitive ratio of the worst-case perturbation. Of all assignments, we could then chose the best one. Such an approach might result in an exponential running time: not only would one have to compute the optimum offline perturbed schedule in order to draw a comparison, which requires to solve an $\mathcal{NP}$-hard problem; more than this, such an algorithm must also determine how the worst-case perturbation looks like (which might be far from trivial). In an online setting, an exponential-time approach is nevertheless applicable.

In this section, we consider a similar approach for bounded perturbations increasing the processing time of at most one job x-fold. The general idea of the algorithm, which we call NoRisk, is as follows. Each time a new job j of the online sequence is presented, the algorithm computes the *worst-case load* of each machine as follows: for a specific machine, it computes the load with the current assignment and given that j is assigned to it, and increases this load with the maximum increase in processing time resulting from perturbing any job assigned to it (i.e., the load if a worst-case perturbation in terms of additional processing time occurs). The algorithm assigns the job to the machine having least worst-case load. Thus, NoRisk is a greedy algorithm, and in each step minimizes the worst-case makespan should the sequence of jobs stop at that point, and the biggest job be perturbed. Here, the perturbation of the biggest job is seen as the perturbation which harms the schedule most. In terms of the competitive ratio this assumption is not valid, but serves as a simple greedy approach that does not require the computation of the optimal offline schedule for each assignment and perturbation.

We now describe NoRisk precisely. We number the jobs in the online sequence increasingly. Thus, the sequence is $J = \{1, \ldots, n\}$. As specified earlier, we assume that at most one job is perturbed. The perturbation increases the job's processing time from p to $p^{\uparrow} = x \cdot p$, for $x > 1, x \in \mathbb{Q}_0^+$. Let $J_i(j), i \in M, j \in J$ be the set of jobs which have already been assigned to machine i when the job $j \in J$ is presented, but j itself has not yet been assigned. Thus, $J_i(1) = \emptyset, \forall i \in M$, and $\bigcup_{i \in M} J_i(n) = J \setminus \{n\}$. We refer to the worst-case load of machine $i \in M$ when NoRisk is considering job j as $\ell_i(j)$. The worst-case load $\ell_i(j)$ is defined as $\ell_i(j) = \sum_{k \in J_i(j)} p_k + p_j + \delta_i^j$, and $\delta_i^j = (x-1) \cdot \max\{p_k | k \in J_i(j) \cup \{j\}\}$ is the worst-case increase in processing time on machine i if job j is assigned to that machine. Abusing notation slightly, we refer to the jobs assigned to machine $i \in M$ when the complete sequence has been scheduled as $J_i(n+1)$ and to the worst-case load of a machine $i \in M$ given the assignment $J_i(n+1)$ as $\ell_i(n+1)$. The algorithm NoRisk is specified precisely in Algorithm 1 using the notation introduced above.

Input : The online sequence of jobs $J = \{1, \ldots, n\}$ each with processing time p_j, $j \in J$.
Output: An online schedule assigning each job to a machine

Initialization:
foreach $i \in M$ **do** $J_i(1) \leftarrow \{\}$;
Online sequence:
for $j \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** m **do**
    compute $\ell_i(j)$;
  end
  Determine machine with least worst-case load:
  $k \leftarrow \texttt{argmin}_{i \in M}(\ell_i(j))$;
  Update assignments of jobs to machines:
  foreach $i \in M$ **do** $J_i(j+1) \leftarrow J_i(j)$;
  $J_k(j+1) \leftarrow J_k(j) \cup \{j\}$;
end

Algorithm 1: The Graham-like risk-aversion online algorithm NoRisk

Before analyzing the algorithm, we remark that, in general, we would be happy with an algorithm which has a worse competitive ratio than Graham's algorithm if no perturbation occurs, given that the competitive ratio is better than Graham's if a perturbation does indeed occur. In the following, we show the competitive ratio of NoRisk if no perturbation occurs.

Theorem 17 *Consider the instances of online scheduling on $m \geq 2$ parallel machines where the processing time of one job may increase to a factor $x \geq 2, x \in \mathbb{Q}_0^+$. Restricted to these instances, and if no perturbation occurs, the* NoRisk *scheduling algorithm has a competitive ratio of x.*

Theorem 17 implies that for $x = 2$, NoRisk is almost as good as Graham's algorithm if no perturbations occur.

Proof Let $\mathcal{L}_{\text{risk}}$ be the makespan obtained by NoRisk. We distinguish two cases. For the first, assume the job $\bar{j}_\mu$ that attains the makespan is scheduled on a machine μ that by removing $\bar{j}_\mu$ is the machine with least load. In this case, the same analysis as for Graham's algorithm applies, since the idle times of the machines other than μ are upper bounded by $p_{\bar{j}_\mu}$, μ has no idle time and the sum of the processing times are a lower bound for m times the optimum value.

For the second case, we assume that the job $\bar{j}_\mu$ that attains the makespan $\mathcal{L}_{\text{risk}}$ on machine μ is scheduled in such

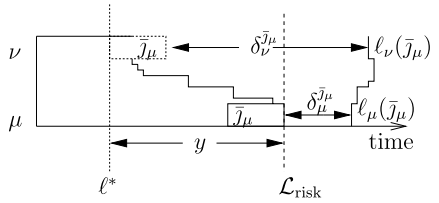

Fig. 7 The setting for the second case of the proof of Theorem 17. The picture reflects the situation when NoRisk was scheduling job $\bar{j}_\mu$. For convenience, machines are sorted according to increasing load

a way that by removing it from the schedule, μ is not the machine with least load. Hence, when μ starts processing $\bar{j}_\mu$, there is at least one machine which has been idle for some time. This situation is shown in Fig. 7. Since $\bar{j}_\mu$ was scheduled on μ, all other machines had a greater (or equal) worst-case load when NoRisk considered $\bar{j}_\mu$. Thus, $\ell_i(\bar{j}_\mu) \geq \ell_\mu(\bar{j}_\mu), i \in M \setminus \{\mu\}$. Now, let ν be a machine having minimum load ℓ^* when NoRisk considered job $\bar{j}_\mu$: $\ell^* = \sum_{k\in J_\nu(\bar{j}_\mu)} p_k \leq \sum_{k\in J_i(\bar{j}_\mu)} p_k, i \in M \setminus \{\nu\}$. Clearly, $\ell^* \leq \mathcal{L}_{\text{OPT}}$, since all machines have load at least ℓ^* when removing $\bar{j}_\mu$. Let $y = \mathcal{L}_{\text{risk}} - \ell^*$. By showing $y \leq (x-1)\mathcal{L}_{\text{OPT}}$ the theorem follows. Since $\bar{j}_\mu$ is the last job scheduled on μ, we have that $\ell_\mu(n+1) = \ell_\mu(\bar{j}_\mu)$. Now,

$$\ell_\mu(\bar{j}_\mu) = \sum_{j\in J_\mu(n+1)} p_j + \delta_\mu^{\bar{j}_\mu} = \mathcal{L}_{\text{risk}} + \delta_\mu^{\bar{j}_\mu} = \ell^* + y + \delta_\mu^{\bar{j}_\mu}.$$

Similarly, $\ell_\nu(\bar{j}_\mu) = \sum_{j\in J_\nu(\bar{j}_\mu)} p_j + p_{\bar{j}_\mu} + \delta_\nu^{\bar{j}_\mu} = \ell^* + p_{\bar{j}_\mu} + \delta_\nu^{\bar{j}_\mu}$. Since the perturbations are bounded by a factor x of the original processing time, $\delta_i^{n+1} \leq (x-1)\cdot\max_{j\in J}\{p_j\} \leq (x-1)\mathcal{L}_{\text{OPT}}, i \in M$. Finally, by definition, $\delta_i^{\bar{j}_\mu} \geq (x-1)p_{\bar{j}_\mu}$. To conclude, let us assume that $y > (x-1)\cdot\mathcal{L}_{\text{OPT}}$. Then,

$$\begin{aligned}\ell_\nu(\bar{j}_\mu) &= \ell^* + p_{\bar{j}_\mu} + \delta_\nu^{\bar{j}_\mu} \leq \ell^* + p_{\bar{j}_\mu} + (x-1)\mathcal{L}_{\text{OPT}}\\ &\leq \ell^* + (x-1)p_{\bar{j}_\mu} + (x-1)\mathcal{L}_{\text{OPT}}\\ &< \ell^* + (x-1)p_{\bar{j}_\mu} + y \leq \ell^* + y + \delta_\mu^{\bar{j}_\mu} = \ell_\mu(\bar{j}_\mu),\end{aligned}$$

a contradiction to having scheduled job $\bar{j}_\mu$ on machine μ. □

An example which comes close to the stated bound if no perturbation occurs is as follows. The online sequence is built by $m-1$ big jobs of processing time 1, each of which is scheduled on an idle machine. These jobs are followed by a sequence of $\frac{x}{y}+1-x$ small jobs of processing time $y < 1$ followed by a small job of processing time $y-\varepsilon$, for an arbitrary small $\varepsilon < y, \varepsilon \in \mathbb{Q}_0^+$. All these jobs are scheduled on the same machine, since the worst-case load of a machine scheduling a big job is $x+y$, whereas the remaining machine not scheduling big jobs has a worst-case load of $ky + \frac{x-1}{y}$ when scheduling the kth small job, which is strictly smaller than $x+y$ for the whole sequence of small jobs. This schedule has a makespan of $\mathcal{L}_{\text{risk}} = x+2y-xy-\varepsilon$, whereas the optimum offline schedule has a makespan of $1+\frac{1}{m}(x+2y-1-xy)$. For $y \to 0$, $m \to \infty$, $\varepsilon \to 0$, and $x \ll \frac{1}{y}$, the ratio of the makespans converges to $\frac{x+2y-xy}{1+\frac{1}{m}(x+2y-1-xy)} \to x$.

Theorem 18 *Consider the instances of online scheduling on $m \geq 2$ identical parallel machines where the processing time of one job may increase to a factor $x \geq 2$, $x \in \mathbb{Q}_0^+$ of its original processing time. Restricted to these instances,* NoRisk *has a competitive ratio of* $1+x-\frac{1}{x}$.

Proof We bound the makespan $\mathcal{L}_{\text{risk}}^{\uparrow}$ of the perturbed instance by bounding the maximum worst-case load, which is an upper bound for any perturbed makespan. Let μ be the machine such that $\ell_\mu(n+1)$ is maximum. By Theorem 17, $\sum_{j\in J_\mu(n+1)} p_j \leq x\mathcal{L}_{\text{OPT}}$ and

$$\begin{aligned}\mathcal{L}_{\text{risk}}^{\uparrow} \leq \ell_\mu(n+1) &\leq \sum_{j\in J_\mu(n+1)} p_j + \delta_\mu^{n+1}\\ &\leq x\mathcal{L}_{\text{OPT}} + (x-1)\max\{p_j : j \in J_\mu(n+1)\}\\ &\leq x\mathcal{L}_{\text{OPT}} + \frac{(x-1)}{x}\max\{p_j^{\uparrow} : j \in J_\mu(n+1)\}\\ &\leq x\mathcal{L}_{\text{OPT}}^{\uparrow} + \frac{x-1}{x}\mathcal{L}_{\text{OPT}}^{\uparrow} \leq \left(1+x-\frac{1}{x}\right)\mathcal{L}_{\text{OPT}}^{\uparrow}.\end{aligned}$$ □

The analysis shows that NoRisk performs worse than Graham's algorithm, which by comparison is slightly better than 3 competitive in this setting. The main handicap is caused by the offset which is introduced when scheduling the unperturbed instance.

An example of a bad instance and perturbation for integral $x \in \mathbb{N}$ is as follows: The online sequence starts with $m-1$ jobs of processing time 1. NoRisk schedules each of these jobs on an idle machine. Then, a sequence of $mx^2+m-xm-1$ small jobs of processing time $\frac{1}{xm}$ is presented, followed by one small job of processing time $\frac{1}{xm}-\varepsilon$, for $\varepsilon < \frac{1}{xm}$, $\varepsilon \in \mathbb{Q}_0^+$. All jobs are scheduled on the same machine μ. Indeed, the worst-case load of μ when the last small job is considered is $(mx^2+m-xm)\frac{1}{xm} + (x-1)\frac{1}{xm} - \varepsilon < x + \frac{1}{x} - 1 + \frac{1}{m} - \frac{1}{xm} \leq x$ for $m \geq 2$ and $x \geq 2$, which is less than the worst-case load $x+\frac{1}{xm}$ or $x+\frac{1}{xm}-\varepsilon$ of the other machines. The sequence finishes with one job of processing time $\frac{1}{x}$. This job is also scheduled on μ, since the worst-case load is $(mx^2+m-xm)\frac{1}{xm} - \varepsilon + \frac{1}{x} + (x-1)\frac{1}{x} = x + \frac{1}{x} - 1 - \varepsilon + 1 = x + \frac{1}{x} - \varepsilon$, compared with the worst-case load $x+\frac{1}{x}$ of the other machines. The worst-case perturbation increases the processing time of the last job in the sequence from $\frac{1}{x}$ to 1, which results in a perturbed makespan $\mathcal{L}_{\text{risk}}^{\uparrow} = x + \frac{1}{x} - \varepsilon$. The optimum offline solution schedules

the m jobs of processing time 1 (one of which is the perturbed job) on different machines, and assigns $x^2 + 1 - x$ small jobs to each machine. In this way, the optimum offline solution has a makespan of $1 + \frac{x}{m} + \frac{1}{xm} - \frac{1}{m}$. For arbitrarily large $m \to \infty$ and arbitrarily small $\varepsilon \to 0$, the ratio of the objectives converges to $\frac{x+\frac{1}{x}}{1+\frac{x}{m}+\frac{1}{xm}-\frac{1}{m}} \to x + \frac{1}{x}$.

We remark the following aspects of NoRisk. First, for a subset of instances, NoRisk produces the same schedule as Graham's algorithm. This happens if the jobs are presented in order of increasing processing time, since in this case, for the jth job in the sequence, all $\delta_i^j, i \in M$, are the same. Thus, the job is scheduled on the least loaded machine. Note that it is on these instances that Graham's algorithm achieves its worst-case competitive ratio. Also, for $x = 1$, NoRisk is Graham's algorithm. Even for $x = 2$, where NoRisk has almost the same competitive ratio as Graham's algorithm, the previous example shows that the latter can perform better than NoRisk. Furthermore, examples can be constructed where the increase of the job with biggest processing time causes a greater makespan to NoRisk than to Graham's algorithm: For example, on two machines and for $\varepsilon < \frac{1}{6}, \varepsilon \in \mathbb{Q}_0^+$, the sequence of jobs with processing times $(1, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6} - \varepsilon, \frac{1}{2}, \frac{1}{2}, 1, 1 + \frac{\varepsilon}{3})$ is such a case, since the perturbation of the last (and biggest) job results in a makespan of $3.5 + \frac{2}{3}\varepsilon$ for Graham's algorithm and in a makespan $4 - \frac{1}{3}\varepsilon$ for NoRisk.

These results show that a greedy approach on the makespan is not what we should aim for. The greatest handicap of this approach is that if no perturbations occur, the machine attaining the makespan can have (unperturbed) load nearly as high as the worst-case load of any machine.

Acknowledgements We thank the anonymous referees for their helpful comments on the presentation of the paper, and for their comments about strengthening Theorems 6 and 15. We also thank Matúš Mihalák for interesting discussions.

References

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: theory, algorithms and applications*. New York: Prentice Hall.

Albers, S. (2002). On randomized online scheduling. In *STOC '02: Proceedings of the 34th annual ACM symposium on theory of computing* (pp. 134–143). New York, NY, USA, 2002. New York: ACM Press. ISBN 1-58113-495-9.

Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Schafer, G., & Vredeveld, T. (2006). Average-case and smoothed competitive analysis of the multilevel feedback algorithm. *Mathematics of Operations Research*, *31*(1), 85–108.

Ben-Tal, A., & Nemirovski, A. (2002). Robust optimization—methodology and applications. *Mathematical Programming*, *92*(3), 453–480.

Chvátal, V. (1983). *Linear programming*. New York: Freeman.

Fleischer, R., & Wahl, M. (2000). Online scheduling revisited. *Journal of Scheduling*, *3*(6), 343–353. Special issue on approximation algorithms for scheduling algorithms (part 2).

Graham, R. L. (1966). Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, *45*(9), 1563–1581.

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, *17*(2), 416–429.

Hall, N. G., & Posner, M. E. (2004). Sensitivity analysis for scheduling problems. *Journal of Scheduling*, *7*, 49–83.

Kasperski, A., & Zieliński, P. (2006). An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, *97*(5), 177–180. ISSN 0020-0190.

Kolen, A. W. J., Rinnooy Kan, A. H. G., van Hoesel, C. S. M., & Wagelmans, A. P. M. (1994). Sensitivity analysis of list scheduling heuristics. *Discrete Applied Mathematics*, *55*, 145–162.

Kouvelis, P., & Yu, G. (1997). *Robust discrete optimization and its applications*. Dordrecht: Kluwer Academic.

Mauroy, G., Wardi, Y., & Proth, J. M. (1997). Sensitivity analysis of machine schedules with multi-priority job classes. In *Proceedings of the 36th IEEE conference on decision and control* (Vol. 5067, pp. 686–691).

Megow, N., Uetz, M., & Vredeveld, T. (2006). Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research*, *31*(3), 513–525.

Möhring, R. H., Schulz, A. S., & Uetz, M. (1999). Approximation in stochastic scheduling: the power of LP-based priority policies. *Journal of the ACM*, *46*(6), 924–942.

Montemanni, R., & Gambardella, L. M. (2004). An exact algorithm for the robust shortest path problem with interval data. *Computers & Operations Research*, *31*, 1667–1680.

Moukrim, A., Sanlaville, E., & Guinan, F. (2003). Parallel machine scheduling with uncertain communication delays. *RAIRO Operations Research*, *37*, 1–16.

Penz, B., Rapine, C., & Trystram, D. (2001). Sensitivity analysis of scheduling algorithms. *European Journal of Operational Research*, *134*, 606–615.

Pinedo, M. (2002). *Scheduling: Theory, algorithms, and systems*. New York: Prentice Hall.

Sanlaville, E. (2005). Sensitivity bounds for machine scheduling with uncertain communication delays. *Journal of Scheduling*, *8*(5), 461–473.

Schäfer, G. (2004). *Worst case instances are fragile. Average case and smoothed competitive analysis of algorithms*. Ph.D. thesis, Universität des Saarlandes, April 2004.

Scharbrodt, M., Schickinger, T., & Steger, A. (2006). A new average case analysis for completion time scheduling. *Journal of the ACM*, *53*(1), 121–146.

Sgall, J. (1998). On-line scheduling. In A. Fiat & G. J. Woeginger (Eds.), *Lecture notes in computer science: Vol. 1442. Online algorithms: The state of the art* (pp. 196–231). Berlin: Springer.